

# AmpliFi → Holder Dividend Engine Zero-Regression Rework Playbook

This document is a surgical, execution-grade rework playbook designed to transform the existing AmpliFi application into a Holder Dividend Engine with near-zero regression risk. Every step explicitly prioritizes reuse of working infrastructure, minimal diff surface area, and deterministic rollout to avoid hours of debugging or emergency fixes.

## 1. Non-Negotiable Rework Constraints

- 1 DO NOT refactor existing working payout, wallet, or claim logic.
- 2 All changes must be additive or disabling, never destructive.
- 3 Every new system must run in shadow mode before activation.
- 4 Feature flags must gate all new logic.

## 2. Inventory of Existing Systems (Must Be Preserved)

- 1 Wallet authentication + session persistence.
- 2 Treasury / reward vault signing utilities.
- 3 Claim window validation & replay protection.
- 4 Payout execution & batching code.
- 5 Admin role permissions & emergency pause.

## 3. Safe Decommissioning Plan (Zero Breakage)

- 1 Disable social ingestion via feature flag, not deletion.
- 2 Freeze campaign tables but do not drop schemas.
- 3 Stub engagement scoring with no-op returns.
- 4 Leave APIs intact to avoid dependency cascades.

## 4. New Weekly Epoch Lifecycle (Critical Path)

- 1 Epoch creation (weekly cron).
- 2 Holder snapshot capture.
- 3 Snapshot finalization lock.
- 4 Ranking + weight calculation.
- 5 Claim window opening.
- 6 Claim window closing.
- 7 Epoch settlement + rollover.

## 5. Snapshot System – Regression-Safe Implementation

- 1 Snapshot runs READ-ONLY against chain/indexer.
- 2 Never mutates balances or wallets.
- 3 Snapshot results stored with immutable epoch\_id.
- 4 Previous snapshots remain queryable.

Schema extension (additive only):

epoch\_id | wallet | balance | first\_seen | snapshot\_time

## 6. Holder Qualification Logic

- 1 Exclude contracts, known LPs, and program wallets.
- 2 Optional: exclude wallets with sell events in epoch.
- 3 Enforce minimum balance threshold (configurable).

## 7. Ranking & Weighting (Deterministic)

Ranking must be fully deterministic and replayable.

- 1 Primary sort: holding duration.
- 2 Secondary sort: balance.
- 3 Cap selection at top 50.
- 4 Persist ranking snapshot.

Weight formula (configurable constant):

$\text{weight} = (\text{holding\_days} ^ \alpha) \times (\text{balance} ^ \beta)$

## 8. Distribution Engine (No New Risk)

- 1 Reuse existing payout execution paths.
- 2 Only input changes: recipient list + amounts.
- 3 Simulate full distribution before enabling claims.

## 9. Claim Window – Hard Guarantees

- 1 Claim eligibility tied to (epoch\_id + wallet).
- 2 One claim per wallet enforced at DB level.
- 3 Nonce invalidation after claim.
- 4 Claims disabled automatically after window closes.

## 10. Unclaimed Funds Policy

- 1 Default: rollover to next epoch.
- 2 Optional: return to creator vault.

- 3 Optional: burn mechanism.

## 11. Admin Control Surface (Minimal Changes)

- 1 Epoch creation override.
- 2 Snapshot re-run (read-only).
- 3 Claim window open/close.
- 4 Emergency pause.

## 12. Frontend Migration Strategy

- 1 Hide unused pages instead of deleting.
- 2 Replace engagement dashboards with leaderboard.
- 3 Reuse claim UI with new data source.
- 4 Feature flag rollout for new views.

## 13. Pre-Launch Checklist (MANDATORY)

- 1 Shadow epoch with zero funds.
- 2 Verify top 50 selection accuracy.
- 3 Verify claim attempts (success + failure).
- 4 Verify admin emergency pause.
- 5 Verify double-claim prevention.

## 14. Launch Sequence (No Surprises)

- 1 Deploy with feature flags OFF.
- 2 Run first funded epoch in parallel dry-run.
- 3 Manually inspect distribution.
- 4 Enable claim window.

## 15. Post-Launch Invariants

- 1 Each epoch is immutable after close.
- 2 No retroactive edits.
- 3 All payouts traceable.
- 4 No dependency on social data.

## 16. Final Outcome

This rework produces a minimal, high-reliability Holder Dividend Engine that preserves AmpliFi's proven infrastructure while eliminating volatility, complexity, and operational risk. The system is intentionally boring — and therefore extremely hard to break.